Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2016

Práctica No. 4 (Tiempo de Ejecución)

Ej. 1. Sean $f(n), g(n), t(n), s(n)$ funciones crecientes no negativas, probar las siguientes propiedades:

- $f(n) \in O(g(n))$ y $t(n) \in O(s(n))$ entonces $f(n) * t(n) \in O(g(n) * s(n))$,
- Probar que la relación $f(n) \equiv_\Theta g(n)$ definida como $f(n) \in \Theta(g(n))$ es una relación de equivalencia.
- Sean $a, b > 0$ dos constantes, $(n + a)^b \in \Theta(n^b)$.
- Probar que cualquier polinomio: $a_d * x^d + a_{d-1} * x^{d-1} + \cdots + a_0 \in O(x^d)$.

Ej. 2. Determinar el tiempo de ejecución, considerando el peor caso, para los siguientes fragmentos de código:

a)
```
for (i = 0; i < n - 1; i++)
                    for (j = i + 1; j < n; j++) {
                      tmp = AA[i][j];
                      AA[i][j] = AA[j][i];
                      AA[j][i] = tmp;
    }
```

b)
```
public static int binarySearch(int[] A, int K){
          int low = 0;
          int hight = A.length -1;
          while (low <= high){
                    int mid = (low + high) /2;
                    if(A[mid]<K) low = mid +1;
                    else if (A[mid]>K) high= mid -1;
                    else return mid;
          }
          return A.length;
   }
```

Ej. 3. Considere el siguiente fragmento de una implementación clásica de listas sobre arreglos:

```
public class ListaSobreArreglos<T extends Comparable>{
        private static final int MAX_LIST = 100;
        private T item[];
        private int numItems;
...
```

Además de las operaciones básicas se desea proveer una operación `public Integer buscar(T x)`, que retorna la posición del elemento en caso de que pertenezca, y una excepción en caso contrario. Está operación debe ser eficiente, se requiere que sea $O(1)$ si el elemento buscado está al final de la lista, y que sea $O(log\ N)$ en otro caso, manteniendo la inserción y la eliminación en $O(N)$.

Ej. 4. Probar que se cumple $log_k\ n \in O(log_t\ n)$, donde k y t son números positivos.

Ej. 5. Dar las ecuaciones de recurrencias para las siguientes funciones en Haskell, y calcular su tiempo de ejecución.

```
maximum :: (Ord a) => [a] -> a
maximum [x] = x
maximum (x:xs)
    | x > maxTail = x
    | otherwise = maxTail
    where maxTail = maximum xs

isPalindrome xs = xs == reverse  xs

reverse [] = []
reverse [x] = [x]
reverse (x:xs) = xs ++ [x]

sort [] = []
sort (x:xs) = insert  x  (sort xs)

insert  x [] = [x]
insert  x (y:ys) = if  x <= y  then  x : (y : ys)  else  y:(insert  x  xs)
```

Ej. 5. Calcular una cota para la siguiente ecuación de recurrencia:

- $T(1) = 1$
- $T(2) = 1$
- $T(n) = T(n/2) + c * n$

Ej. 6 Demostrar $log\, n \in O(n)$ y $log\, n \in O(\sqrt{n})$

Ej. 7. Resuelva las siguentes ecuaciones de recurrencia:

- $T(1) = 1$, $T(n) = 2 * T(n/2) + n$.
- $T(1) = 1$, $T(n) = 9 * T(n/3) + n$.